Discrete Minds in a Continuous World: Do Language Models Know Time Passes?

Minghan Wang, Ye Bai, Thuy-Trang Vu, Ehsan Shareghi, Gholamreza Haffari
Department of Data Science & AI, Monash University
{firstname.lastname}@monash.edu

Abstract

While Large Language Models (LLMs) excel at temporal reasoning tasks like event ordering and duration estimation, their ability to perceive the actual passage of time remains unexplored. We investigate whether LLMs perceive the passage of time and adapt their decision-making accordingly through three complementary experiments. First, we introduce the Token-Time Hypothesis, positing that LLMs can map discrete token counts to continuous wall-clock time, and validate this through a dialogue duration judgment task. Second, we demonstrate that LLMs could use this awareness to adapt their response length while maintaining accuracy when users express urgency in question answering tasks. Finally, we develop BombRush, an interactive navigation challenge that examines how LLMs modify behavior under progressive time pressure in dynamic environments. Our findings indicate that LLMs possess certain awareness of time passage, enabling them to bridge discrete linguistic tokens and continuous physical time, though this capability varies with model size and reasoning abilities. This work establishes a theoretical foundation for enhancing temporal awareness in LLMs for time-sensitive applications.[1]

1 Introduction

Current LLMs are predominantly employed for **time-insensitive** tasks such as question answering (QA) (Hendrycks et al., 2021; Rein et al., 2023), summarization (Zhang et al., 2023), and translation (Xu et al., 2024). Although both user input and LLM generation inherently consume time, real-world temporal factors typically do not significantly influence these tasks' outcomes. For instance, neither the duration taken by a user to formulate a question nor the LLM's response latency substantially alters the correctness of the final output.

[1]Dataset and code are available at https://github.com/yuriak/LLMTimePerception

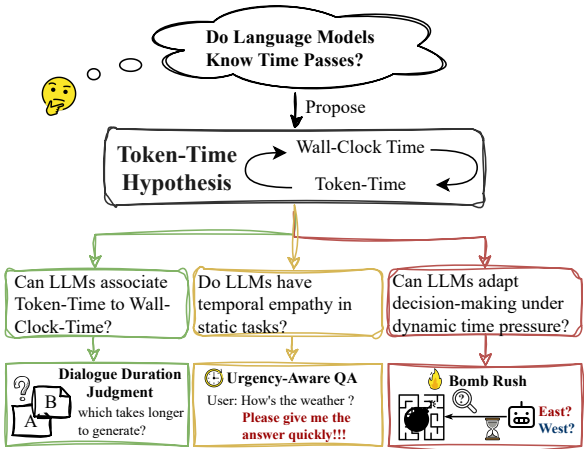

Figure 1: Overview of our work. We propose the Token-Time Hypothesis connecting discrete tokens to continuous time, then validate it through three experiments examining time-mapping capabilities, temporal empathy in static tasks, and adaptive decision-making under dynamic tasks.

However, numerous **time-sensitive** tasks exist in real-world scenarios, including simultaneous translation (Wang et al., 2024b,a), autonomous driving (Yang et al., 2024b), real-time dialogue (Défossez et al., 2024), and robotic control (Driess et al., 2023). In these contexts, environmental conditions continually evolve over time, resulting in input data variability and task dynamics heavily dependent on temporal progression. Moreover, the time taken by a model to generate responses can critically impact task outcomes. Hence, temporal progression becomes a decisive factor in these scenarios. This raises a fundamental question: *Are LLMs capable of perceiving and interpreting the passage of time? Specifically, can LLMs recognize their temporal context and adapt their behavior accordingly?*

To date, extensive research has focused on LLMs' reasoning capabilities in time-related tasks, such as event ordering (Zhou et al., 2021; Tan et al., 2023; Ding and Wang, 2025), temporal expression parsing (Chen et al., 2021; Zhang and Choi,

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18703–18729
November 4-9, 2025

2021; Zhou et al., 2019), and duration computation (Wang and Zhao, 2024; Jia et al., 2018; Shang et al., 2022; Mavromatis et al., 2021). These studies evaluate models' ability to perform temporal reasoning **but do not address whether models possess any awareness of the passage of time itself within our physical world**.

In this paper, we bridge this gap by investigating whether and how LLMs perceive the passage of time and adapt their behavior accordingly. We first analyze the mechanisms through which LLMs might understand time and propose the **Token-Time Hypothesis**: LLMs can establish connections between text length and real-world temporal progression. Through three complementary experiments, we systematically explore the nature and boundaries of this capability.

Our investigation begins with a Dialogue Duration Judgment task that validates the Token-Time Hypothesis by examining whether LLMs can accurately determine which conversation took longer based on different temporal cues. We find that models can indeed associate token count with elapsed time, though their ability to reconcile conflicting cues depends significantly on their reasoning capabilities. Building on these insights, we then explore how this temporal awareness manifests in practical scenarios through an Urgency-Aware QA experiment, which reveals that models strategically reduce response length when users express time constraints while preserving answer quality, demonstrating a form of "temporal empathy." Finally, our BombRush experiment extends this investigation to dynamic, interactive environments, showing how models adjust their reasoning depth and decision-making strategies under progressively increasing time pressure.

This work makes several key contributions to our understanding of language model capabilities: **(1)** We present the first comprehensive study investigating LLMs' perception of temporal progression. **(2)** We establish the Token-Time Hypothesis as a theoretical framework explaining how discrete linguistic tokens might be mapped to continuous temporal experience. **(3)** We design three novel experimental paradigms to evaluate how time passage perception influences LLM behavior across diverse scenarios. **(4)** We provide empirical evidence that LLMs possess certain temporal passage awareness, though varying across model scales and genres. These findings have important implications for deploying LLMs in time-sensitive applications and provide a foundation for future research on enhancing temporal awareness in language models.

2 Related Work

Temporal Understanding and Reasoning

Temporal understanding involves identifying the time scope (e.g., start and end time) of events in text, which often requires commonsense inference—for example, interpreting "during the Second World War" as referring to the period from 1939 to 1945. Temporal reasoning, in contrast, refers to deducing the temporal relationship (e.g., before, after, between) between events or between the query and text (Chen et al., 2021).

Recent studies have investigated whether LLMs possess basic temporal commonsense. Jain et al. (2023) benchmark LLMs across multiple datasets and prompting strategies, revealing limited and inconsistent performance across tasks like event ordering and duration reasoning. Zhang et al. (2024) propose TIMEARENA, a simulated environment to test LLMs under temporal constraints, showing that models struggle with effective time-sensitive planning. Thukral et al. (2021) evaluate models on NLI tasks involving temporal expressions and find weaknesses in understanding time-related comparisons. Corresponding benchmark datasets in this area include TimeDial (Qin et al., 2021), TNLI (Hosokawa et al., 2023), and MC-TACO (Zhou et al., 2019), which focus on evaluating temporal inference, commonsense alignment, and fine-grained question answering involving time-related knowledge.

In parallel, several studies aim to enhance temporal understanding by introducing time-aware training strategies. Kimura et al. (2022) use multi-step fine-tuning and masked temporal indicators to improve commonsense inference across time-related tasks. TSM (Cole et al., 2023) propose Temporal Span Masking, targeting time expressions during pretraining to strengthen temporal representations. BiTimeBERT (Wang et al., 2023) incorporates bi-temporal signals from news corpora to build time-sensitive language models.

Time Sensitive Question Answering

A closely related line of work focuses on Time-Sensitive Question Answering (TSQA), which refers to answering questions where the response depends critically on a specific time reference—changing the time expression leads to a different answer (Chen et al., 2021). Solving

TSQA tasks requires both temporal reasoning and commonsense understanding to interpret and align facts within a temporal context. Commonly used datasets in this domain include TempReason (Tan et al., 2023), TimeQA (Chen et al., 2021), SituatedQA (Zhang and Choi, 2021).

To improve TSQA performance, prior work has explored methods for enhancing temporal sensitivity through training or model augmentation. Yang et al. (2024a) introduce a framework that combines temporal-aware embeddings with contrastive reinforcement learning to help models better align facts with time-conditioned queries. Similarly, Su et al. (2023b) incorporate temporal graphs into Transformer architectures, allowing the model to reason more effectively over event-time structures.

These prior efforts have made significant progress in improving temporal reasoning, representation, and sensitivity in language models. **However, to the best of our knowledge, no existing work has examined whether LLMs possess any awareness of the passage of time itself within our physical world.** Unlike traditional temporal QA or reasoning tasks, this question shifts focus from reasoning about time to perceiving time as an experiential dimension—raising new challenges at the intersection of cognitive modeling and LLM behavior analysis.

3 LLM's Sense of Time

Time perception (chronoception) in psychology refers to the subjective experience of time: how individuals sense durations and intervals between events, distinct from objective chronological time (Di Lernia et al., 2018; Eagleman, 2008; Wittmann, 2009).

Building on this concept, we investigate whether similar mechanisms exist in AI systems. Autoregressive decoder-only language models process text by decomposing sequences into conditional probability distributions that preserve temporal ordering (Brown et al., 2020). This creates a natural alignment between token positions and chronological progression: earlier tokens precede later ones, mirroring how events unfold in human experience. Moreover, token quantity roughly corresponds to communication time in human interaction. While humans intuitively incorporate temporal dimensions in communication, a fundamental question emerges: do LLMs possess an analogous sense of time that influences their comprehension and generation?

3.1 Time Perception Mechanisms in LLMs

To systematically investigate potential temporal awareness in language models, we focus on the most representative scenario, i.e. multi-turn dialogue, where time perception becomes particularly relevant. Conversations provide an ideal context due to their inherent chronological structure, participants exchange messages sequentially, creating a natural temporal flow that allows us to examine how LLMs perceive and process time-related information.

Within this interactive framework, an LLM's operational cycle can be abstracted into two states: **Standby State** (awaiting input without computation) and **Active State** (processing input through generation completion). During standby, the model exists in temporal isolation, with time perception potentially manifesting only in two scenarios:

1. **During Encoding of User Input**: When activated by new input, the model encodes this text, potentially extracting temporal information from either the number of input tokens or explicit time markers within the input. This mechanism allows the model to infer time that elapsed during its standby period.
2. **During Autoregressive Generation**: While generating text, the model cannot incorporate new external input, but theoretically could estimate passing wall-clock time if it possesses awareness of its own generation speed and the relationship between the number of generated tokens and time elapsed.

3.2 The Token-Time Hypothesis

These observations lead us to formulate the **Token-Time Hypothesis**, which proposes that: *LLMs treat tokens as discrete temporal units, inferring the passage of real-world time from the length and sequencing of textual events within the token space*. This hypothesis establishes two distinct temporal measurement systems: **Token-Time**, the discrete, abstract temporal metric based on token counts, and **Wall-Clock-Time**, the continuous, physical temporal metric in the real world. Under this framework, studying an LLM's temporal awareness fundamentally involves understanding how the model perceives and utilizes the mapping relationship between these two temporal domains. **This mapping capability, if present, would be crucial for LLMs**

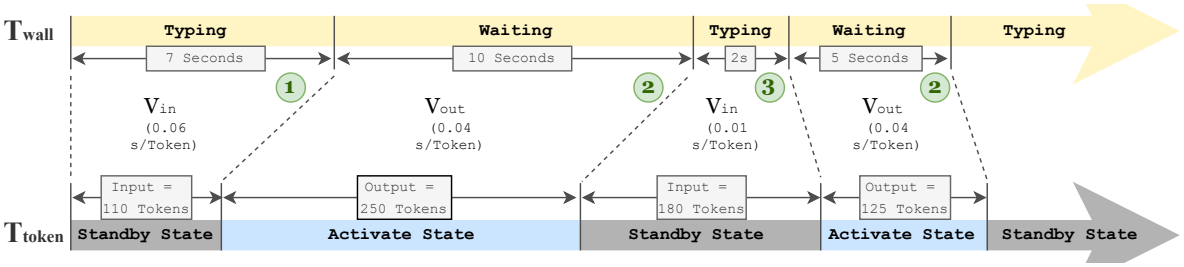

Figure 2: Illustration of the relationship between Token Time and Wall-Clock-Time across LLM's operational cycles. This figure compares the Wall-Clock timeline (T_{wall}) and the model's Token timeline (T_{token}). The LLM alternates between a **Standby State** (waiting for user input) and an **Activate State** (generating responses). While the LLM can observe the number of input tokens, it cannot infer how long the user spent composing them, as input speed is unobservable and variable. For example, in Case ① , the user typed slowly while thinking, yielding few tokens over a long period. In Case ③ , the user quickly pasted a long passage, resulting in many tokens in a short time. In contrast, during the generation phase, the LLM's output token count provides a measurable passage of time, assuming an ideally fixed generation speed (V_{out}), as shown in Case ② . **This enables an estimation of elapsed Wall-Clock Time based on the number of output tokens.**

to function effectively in time-sensitive applications where understanding temporal progression impacts task performance. (More discussions can be found in Section A.2.)

3.3 Mapping Between Token-Time and Wall-Clock-Time

To formalize the relationship between these temporal domains, we define a mathematical framework that quantifies this critical mapping:

$$T^m_{wall} = T^m_{tok} \times V_m, \quad m \in \{in, out\} \tag{1}$$

where V_m represents the conversion rate between domains for mode m (input or output), measured in seconds per token (s/token). Specifically, the input conversion rate V_{in} is calculated as the ratio of Wall-Clock time spent on input preparation T^{in}_{wall} to the total token count of that input T^{in}_{tok}. Similarly, the output conversion rate V_{out} represents the ratio of actual generation time T^{out}_{wall} to the number of tokens produced T^{out}_{tok}.

In ideal conditions, when both V_{in} and V_{out} remain constant, the relationship $T_{wall} \propto T_{tok}$ would hold. However, real-world scenarios demand a more nuanced consideration, particularly for V_{in}. In multi-turn dialogues, various external factors (as illustrated in Figure 2) cause V_{in} to fluctuate significantly, preventing a consistent $T^{in}_{wall} \propto T^{in}_{tok}$ relationship. Conversely, V_{out} remains relatively stable (assuming negligible computational overhead from increasing generation length), maintaining the proportionality $T^{out}_{wall} \propto T^{out}_{tok}$. To ensure experimental rigor, we control for input variability and focus primarily on LLMs' understanding and utilization of time passage during the Activate State (output) phase.

Conversation A
User: How's the weather today?
LLM: It's a sunny day.

Conversation B
User: How's the weather today?
LLM: It's a sunny day with mild temperatures. There are clear skies and warm air, perfect for outdoors.

You are given two conversations (A and B), each with a user prompt and an LLM response. Assuming the same LLM with constant inference speed. **which response took the LLM longer to generate — A or B?**

Figure 3: Prompt example for the Dialogue Duration Judgment task. LLMs are given two user–LLM conversations and asked to judge which response took the LLM longer to generate.

Specifically, to address our research questions, we conducted three experiments. The first experiment validates whether LLMs can comprehend the $T^{out}_{wall} \propto T^{out}_{tok}$ mapping relationship. In the subsequent two experiments, we investigate whether LLMs with time passage perception capabilities demonstrate behavior that strategically controls T^{out}_{tok} to influence T^{out}_{wall}, thereby meeting the temporal requirements of particular tasks.

4 Dialogue Duration Judgment

Our first experiment examines whether LLMs can associate text length (Token-Time) with temporal duration (Wall-Clock-Time), and how this capability functions under various confounding factors. To systematically evaluate this, we designed a **dialogue duration judgment** task using paired samples from the chatbot_arena_conversations dataset[2] (Zheng et al., 2023). Each sample contains

[2]lmsys/chatbot_arena_conversations

Setting	Key Characteristics	Purpose
S1	No explicit temporal cues provided	Test inherent ability to infer temporal duration from T_{tok}
S1-Hint	Textual hint: "Generation time is proportional to number of tokens"	Evaluate if the model understands the hint: $T_{tok} \propto T_{wall}$
S1-Count	Direct token counts provided for both dialogues	Test ability to judge duration from explicit T_{tok} information
S2	Timestamps provided for input and response; consistent with text length	Test prioritization of explicit T_{wall} cues over implicit T_{tok}
S2-M	Misleading timestamps: longer responses shown with shorter durations	Evaluate ability to prioritize T_{wall} over contradictory T_{tok} cues
S2-M+	Both misleading timestamps and accurate token counts provided	Determine which temporal domain (T_{tok} vs T_{wall}) models prioritize when directly conflicting

Table 1: Experimental settings for the Dialogue Duration Judgment task. Settings are divided into Token-Time cues (S1, S1-Hint, S1-Count) and Wall-Clock-Time cues (S2, S2-M, S2-M+).

a user prompt with two responses (A or B) from different LLMs of substantially different lengths. We selected 300 samples where both responses were generated by high-quality models such as GPT-4 or Claude, ensuring response quality would not negatively affect LLMs' decision-making. The tested LLM must identify which dialogue is likely to require more time to complete, and provide justification for its judgment (See Figure 3 for an example).

4.1 Experimental Setup

We systematically designed six controlled settings by introducing different temporal cues that either facilitate or impede the LLM's judgment. These settings are organized into two categories: **Token-Time cues** and **Wall-Clock-Time cues**, each probing whether the model could correctly leverage the corresponding temporal domain to make accurate predictions. The detailed task prompts and demonstration of settings are illustrated in Section B.1, B.2 and Table 1.

Setting 1: Token-time Cues For this group of settings, we progressively introduced more explicit cues to help the candidate LLM establish the connection between Token-Time and Wall-Clock-Time. To eliminate potential confounding factors, we explicitly stated in the prompt that all responses were generated by an identical model with constant inference speed.

Setting 2: Wall-Clock-Time Cues In this group, we introduced temporal logs containing explicit Wall-Clock-Time timestamps for both dialogues to test whether the LLM could effectively utilize these direct temporal signals. We also manipulated these timestamps in certain settings to create conflicting cues, allowing us to observe how LLMs prioritize different temporal indicators. These manipulations simulate realistic scenarios where varying computational conditions might create discrepancies between text length and generation time.

	Settings	S1	S1-Hint	S1-Count	S2	S2-M	S2-M+
SLM	**Llama-8B**	79.7	83.0	**99.4**	76.8	47.7	16.3
	Qwen-7B	66.8	71.2	**92.7**	80.9	26.8	12.9
LLM	**Llama-70B**	91.3	93.2	**99.9**	93.5	74.0	40.1
	Qwen-72B	83.8	84.5	**99.9**	96.6	82.5	48.8
LRM	**DLlama-70B**	87.9	92.3	**100.0**	99.2	98.0	94.3
	QwQ-32B	91.5	94.8	**100.0**	99.3	99.2	99.1

Table 2: Accuracy (%) of different model categories on the Dialogue Duration Judgment task. Settings include Token-Time cues (S1: baseline, S1-Hint: with hint, S1-Count: with token counts) and Wall-Clock-Time cues (S2: consistent timestamps, S2-M: misleading timestamps, S2-M+: misleading timestamps with token counts).

This experimental design probes how LLMs negotiate between different temporal metrics and whether they possess coherent mechanisms for temporal perception.

Models and Evaluation We tested six models divided into three categories: Small LMs (7-8B parameters): Qwen-2.5-7B[3] (Team, 2024) and Llama-3.1-8B[4] (Grattafiori et al., 2024); Large LMs (70B+ parameters): Llama-3.3-70B[5] and Qwen-2.5-72B[6]; and Large Reasoning Models (LRMs with internalized inference time scaling capabilities): DeepSeek-R1-Distill-Llama-70B[7] (DeepSeek-AI, 2025) and QWQ-32B[8] (Team, 2025). For each model under each setting, we conducted 5 replications and reported the average accuracy as our primary evaluation metric. To ensure statistical rigor, we further conducted 20 replications for each model-setting combination, with detailed significance testing in Section B.3.

4.2 Results

Table 2 presents the accuracy of all models across different experimental settings. We analyze performance by model category and within settings,

[3]Qwen/Qwen2.5-7B-Instruct
[4]meta-llama/Llama-3.1-8B-Instruct
[5]meta-llama/Llama-3.3-70B-Instruct
[6]Qwen/Qwen2.5-72B-Instruct
[7]deepseek-ai/DeepSeek-R1-Distill-Llama-70B
[8]Qwen/QwQ-32B

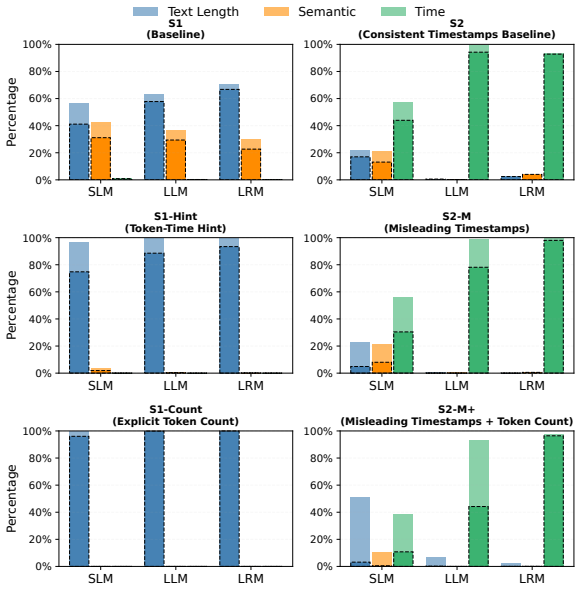

Figure 4: Attribution distribution and accuracy across model classes and settings. Each bar shows the proportion of attribution used, darker segments indicate the percentage of correct answers within that attribution. (See Section B.4 for detailed numerical results)

followed by attribution analysis to understand underlying reasoning mechanisms.

Performance Across Model Categories We observe a clear progression in accuracy from SLMs to LLMs to LRMs, aligning with expectations that larger models and those with enhanced reasoning capabilities demonstrate superior overall performance, potentially including the ability to perceive the passage of time that we are investigating.

Token-Time Settings (S1, S1-Hint, S1-Count) In the baseline condition (S1), all models achieve reasonably high accuracy without explicit cues, indicating an inherent capability to associate text length with temporal duration. As Token-time hints become more explicit, accuracy increases across all model categories, reaching near-perfect scores in S1-Count where token counts are explicitly provided.

Wall-Clock-Time Settings (S2, S2-M, S2-M+) Most models achieve higher accuracy with explicit timestamps (S2) compared to the Token-time baseline (S1), indicating that direct temporal cues enhance duration judgment beyond default token-based heuristics.

The most revealing results emerge with misleading cues (S2-M and S2-M+). SLMs and LLMs show dramatic performance degradation when token counts and timestamps provide contradictory signals. In stark contrast, LRMs maintain remarkably high accuracy even under these challenging conditions, with QwQ-32B achieving 99.1% accuracy in the most conflicting setting.

4.2.1 Attribution Analysis

To understand the underlying mechanisms of temporal judgment, we classified models' justifications into three categories: "text length," "semantic," and "time" using a Llama-3.3-70B classifier (see Figure 9 for the prompt). Figure 4 shows attribution distributions and corresponding accuracy across settings and model categories.

In the baseline setting (S1), all models primarily use text length for attribution, though with considerable reliance on semantic cues as well. Judgments based on text length achieve substantially higher accuracy than those based on semantic features. As Token-time hints become more explicit, models increasingly favor text length attributions over semantic cues. For Wall-Clock-Time settings, LLMs show greater aptitude than SLMs at leveraging timestamp information in S2. Interestingly, LRMs occasionally incorporate multiple attribution types, but mostly rely on timestamp cues. Under misleading settings (S2-M), SLMs and LLMs maintain similar attribution patterns as in S2, but with significantly reduced accuracy. In the most challenging setting (S2-M+), both SLMs and LLMs are substantially misled, increasingly relying on text length despite contradictory timestamp information. Even when LLMs continue using timestamp cues, their accuracy declines dramatically, indicating confusion when faced with conflicting temporal signals. LRMs, however, remain largely resistant to these misleading cues. We speculated more about this phenomenon's underlying mechanism in Section A.3.2.

Key Takeaway

LLMs can perceive the passage of time by connecting Token-Time with Wall-Clock-Time, although this intriguing ability varies with their underlying reasoning capabilities.

5 Urgency-Aware QA

Just as humans demonstrate emotional empathy by resonating with others' feelings (Manzoor et al., 2024), interactive AI systems should exhibit what we might call *temporal empathy*: the ability to rec-

Model		OpenbookQA			GSM8K			GPQA		
		Normal	Urgent	Δ%	Normal	Urgent	Δ%	Normal	Urgent	Δ%
		Accuracy / # Tokens (GPT-4o Tokenizer)								
SLM	Llama-8B	84.6 / 299	84.2 / 302	-0.4% / 0.9%	82.2 / 482	81.9 / 439	-0.4% / -8.9%	28.1 / 2,900	28.3 / 2,851	-1.7% / 0.7%
	Qwen-7B	87.3 / 203	88.2 / 186	1.0% / -8.4%	88.8 / 271	88.6 / 258	-0.2% / -4.9%	33.9 / 598	34.3 / 551	1.2% / -7.8%
LLM	Llama-70B	96.6 / 289	96.4 / 259	-0.1% / -10.4%	93.2 / 242	93.1 / 226	-0.1% / -6.5%	49.0 / 808	49.3 / 754	0.6% / -6.7%
	Qwen-72B	96.4 / 206	96.2 / 164	-0.2% / -20.7%	91.4 / 273	91.3 / 242	-0.1% / -11.4%	48.2 / 695	52.1 / 635	8.2% / -8.6%
LRM	DS-Llama-70B	95.8 / 784	95.9 / 683	0.1% / -12.9%	92.9 / 464	93.8 / 441	1.0% / -5.0%	33.8 / 5,106	38.0 / 5,065	12.2% / -0.8%
	QwQ-32B	96.2 / 793	96.6 / 695	0.4% / -12.3%	96.2 / 759	96.6 / 655	0.4% / -13.7%	59.5 / 6,847	62.4 / 6,332	4.9% / -7.5%

Table 3: Urgent-Aware QA results on OpenbookQA, GSM8K, and GPQA benchmarks. We report the accuracy under **Normal** and **Urgent** prompt conditions, along with relative changes (Δ%). Token usage is measured using the GPT-4o tokenizer to ensure cross-model comparability.

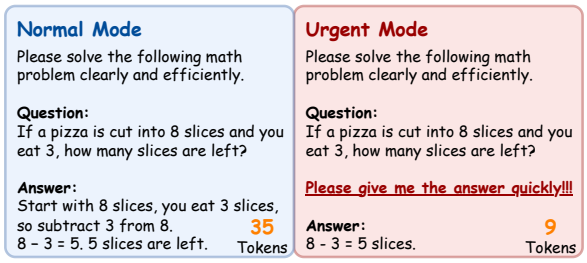

Figure 5: An example of the Urgency-Aware QA task, where the urgency expression is highlighted.

ognize and adapt to a user's perception of time. Building on our finding that LLMs can associate Token-Time with Wall-Clock-Time, we now investigate their ability to respond appropriately to user time constraints. This represents a practical application of time passage perception, where models leverage their understanding of temporal constraints to adapt their behavior. To examine this temporal adaptation, we introduce an urgency-aware QA task that uses static scenarios to observe how models respond under time pressure. This experiment specifically measures changes in response length and accuracy when models are explicitly prompted to answer quickly.

5.1 Experimental Setup

We selected three QA datasets of varying difficulty levels: Openbook-QA (commonsense) (Mihaylov et al., 2018), GSM8k (mathematics) (Cobbe et al., 2021), and GPQA-Diamond (scientific questions) (Rein et al., 2023), totaling 2,017 questions. Each question was presented under two conditions: a **normal mode** and an **urgent mode**. In normal mode, we used a standard prompt asking the LLM to provide step-by-step reasoning and the corresponding answer. In urgent mode, we augmented the prompt with an additional sentence randomly sampled from a pool of urgency expressions (e.g., *"I'm in a big hurry right now. Please give me the answer quickly!!!"*).

Following the methodology of our previous experiment, we tested six models across three categories (SLM, LLM, LRM), conducting 5 replications per model per setting to ensure reliability. We evaluated performance primarily through accuracy and assessed time efficiency through token count. For cross-model comparability, we calculated token counts using both model-specific tokenizers and a shared GPT-4o tokenizer (via tiktoken). Full prompt templates and the urgency pool are provided in Appendix C.2.

5.2 Results

Table 3 presents model performance across three datasets under normal and urgent conditions, showing accuracy, token usage, and corresponding relative changes. Rather than focusing on absolute metrics, we concentrate on the relative changes that reveal models' adaptation to urgency.

Token Usage Adaptation The most consistent pattern across our experiments is the reduction in token usage under urgent conditions. Nearly all models demonstrate decreased token consumption when prompted with urgency expressions, suggesting an inherent ability to associate time pressure with brevity. Qwen-72B exhibits the most consistent token reduction across all datasets, potentially indicating stronger temporal adaptation capabilities. When comparing across datasets of increasing difficulty, we observe that token reduction is more pronounced in simpler tasks (OpenbookQA and GSM8K averaging 10.6% and 8.4% reductions, respectively) compared to the more challenging GPQA (5.5% average reduction). This suggests a deliberate balancing mechanism, i.e., **models recognize the need for brevity under time constraints but appropriately calibrate this reduction according to task complexity.**

Impact on Accuracy For the relatively easy datasets (OpenbookQA and GSM8K), models maintain nearly identical accuracy levels despite using fewer tokens in urgent mode. This demonstrates their capability to preserve reasoning quality while responding to temporal constraints. Surprisingly, on the challenging GPQA dataset, five of six models showed accuracy improvements in urgent mode, with particularly notable gains from Qwen-72B (+8.2%) and both LRMs (DeepSeek-Llama-70B: +12.2%, QwQ-32B: +4.9%). This contradicts the conventional assumption that reasoning chain length positively correlates with accuracy (Wei et al., 2023). We hypothesize that urgency prompting may encourage models to select more efficient reasoning trajectory, potentially reducing opportunities for hallucination by avoiding unnecessary exploratory reasoning. This finding suggests a potential approach for mitigating the overthinking problem in large reasoning models' research (Sui et al., 2025). We also discussed the potential semantic connection of "being quick" and "being brief" in Section A.3.4.

Key Takeaway

LLMs demonstrate temporal empathy by recognizing user time constraints and producing more concise outputs without sacrificing performance, revealing sophisticated balancing of brevity and thoroughness based on task demands.

6 BombRush: Time-pressured Navigation Tasks

Our third experiment examines LLMs in time-sensitive dynamic tasks through BombRush, a grid-world navigation challenge where temporal constraints progressively impact decision-making, extending our investigation from the static QA task to sequential, interactive scenarios.

In BombRush, the LLM functions as an agent navigating an N×N (N=8 in all experiments) grid-world to locate a timed bomb before detonation. The bomb is invisible on the map but emits signals indicating its distance and bearing to guide the agent for navigation. The environment contains visible walls that obstruct movement, requiring the agent to plan routes wisely to avoid obstacles. During simulation, the agent receives an environment state at each step (map representation, wall coordinates, bomb signals and remaining time in seconds) and responds with an action (moving one step in one of four cardinal directions) accompanied by its reasoning text, creating a continuous decision loop.

Crucially, we establish a direct mapping between reasoning token usage (Token-Time) and simulated elapsed time, rather than using real-world time that would introduce variability across models. Through a predetermined output conversion rate V_{out}, the bomb's countdown directly correlates with the model's reasoning length (e.g., 500 tokens consume 5 seconds). This design compels the LLM to recognize that verbose reasoning consumes time, requiring it to balance thoroughness with conciseness to maximize movement opportunities while efficiently approaching the bomb.

6.1 Experimental Setup

We designed three progressive settings with increasing complexity (see detailed comparison in Figure 7):

1. **S1: Treasure Hunt**: The bomb is replaced with a treasure, and the time limit is removed. This setting serves as the baseline to evaluate the model's fundamental spatial navigation capabilities without temporal constraints.
2. **S2: Bomb Rush**: This setting serves as the representative one where the bomb emits directional signals while counting down to detonation.
3. **S3: Bomb Rush Hard**: The most challenging setting: the bomb moves randomly and no longer emits continuous signals. The agent must strategically choose between sacrificing movement to detect the bomb's current position or advancing based on outdated information. This requires strategic balancing of information gathering and movement under time pressure.

For this task, we tested only the LLM and LRM model groups, as we found that SLMs lacked the basic spatial navigation capabilities necessary for the test and therefore excluded them from the experiments. For each model, we run 100 simulations per setting and report results accordingly. All prompts, hyperparameters, and variations of settings are illustrated in Section D.

6.2 Results

Figure 6 shows the relationship between average token usage per step and remaining time across

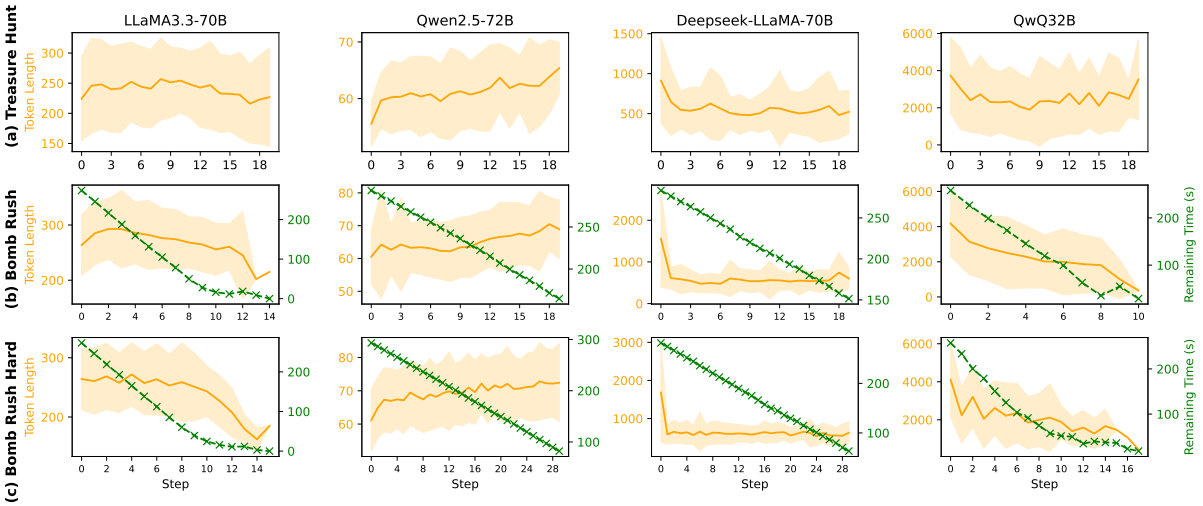

Figure 6: Step-wise token usage and remaining time across three settings. For each model, the orange curve indicates the average number of tokens generated per step, with shaded areas representing standard deviation. The green line (where applicable) shows the average remaining time on that step. This figure illustrates how reasoning token usage evolves over time under different settings.

settings. In Treasure Hunt (no time pressure), models maintain stable token usage. In contrast, during Bomb Rush tasks, Llama and QwQ demonstrate decreasing token usage as time diminishes, with QwQ showing particularly strong adaptation. Qwen-2.5-70B exhibits minimal change due to its consistently concise reasoning style, while Deepseek-distilled-Llama shows a notable token reduction after the first step in all settings, with larger decreases in time-pressured scenarios. More analysis on metrics such as task success rate can be found in Section D.

Key Takeaway

In dynamic environments under increasing time pressure, models adapt behavior, primarily by reducing reasoning verbosity to conserve time. However, inconsistent adaptation patterns across models reveal varying degrees of temporal awareness and its influence on decision-making.

7 Conclusion

We have investigated whether LLMs perceive and adapt to time passage, a capability distinct from temporal reasoning. Through the proposed Token-Time Hypothesis and three complementary experiments, we found that LLMs can, to some extent aware of the correlation between token-time and wall-clock time, indicating an emergent form of temporal awareness. This awareness manifests as strategic behavioral adaptations: e.g. response length adjustment to match urgency or adaptive decision-making under time pressure. Future research might explore enhancing this capability through specialized training, investigating how temporal perception might address known limitations like overthinking.

Limitations

While our work provides novel insights into LLMs' time perception capabilities, several limitations should be acknowledged. Our study primarily establishes the presence and variability of temporal awareness in LLMs but does not provide specific methods to enhance this capability, which is an important next step for improving performance in time-sensitive applications. Additionally, our experiments examine text-only models within controlled settings, whereas real-world applications would face multimodal and variable computational environments affecting the Token-Time to Wall-Clock-Time relationship. In addition, our study primarily tested open-source models, excluding commercial models such as GPT-4o or Claude. More powerful commercial models might lead to conclusions that differ from our current findings. Finally, while we observed substantial differences in time perception capabilities across models, understanding the underlying mechanisms driving these differences remains an area for future research.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. *Preprint*, arXiv:2108.06314.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jeremy R. Cole, Aditi Chaudhary, Bhuwan Dhingra, and Partha Talukdar. 2023. Salient span masking for temporal understanding. *Preprint*, arXiv:2303.12860.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Daniele Di Lernia, Silvia Serino, Giovanni Pezzulo, Elisa Pedroli, Pietro Cipresso, and Giuseppe Riva. 2018. Feel the time. time perception as a function of interoceptive processing. *Frontiers in Human Neuroscience*, Volume 12 - 2018.

Xi Ding and Lei Wang. 2025. Do language models understand time? In *The First International Workshop on Transformative Insights in Multifaceted Evaluation at The Web Conference 2025*.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. Palm-e: An embodied multimodal language model. *Preprint*, arXiv:2303.03378.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. *Preprint*, arXiv:2410.00037.

David M Eagleman. 2008. Human time perception and its illusions. *Current Opinion in Neurobiology*, 18(2):131–136. Cognitive neuroscience.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Taishi Hosokawa, Adam Jatowt, and Kazunari Sugiyama. 2023. Temporal natural language inference: Evidence-based evaluation of temporal text validity. In *Advances in Information Retrieval*, pages 441–458, Cham. Springer Nature Switzerland.

Raghav Jain, Daivik Sojitra, Arkadeep Acharya, Sriparna Saha, Adam Jatowt, and Sandipan Dandapat. 2023. Do language models have a common sense regarding time? revisiting temporal commonsense reasoning in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6750–6774, Singapore. Association for Computational Linguistics.

Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1807–1810. ACM.

Mayuko Kimura, Lis Kanashiro Pereira, and Ichiro Kobayashi. 2022. Toward building a language model for understanding temporal commonsense. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 17–24, Online. Association for Computational Linguistics.

Muhammad Arslan Manzoor, Yuxia Wang, Minghan Wang, and Preslav Nakov. 2024. Can machines resonate with humans? evaluating the emotional and empathic comprehension of lms. *Preprint*, arXiv:2406.11250.

Costas Mavromatis, Prasanna Lakkur Subramanyam, Vassilis N. Ioannidis, Soji Adeshina, Phillip R. Howard, Tetiana Grinberg, Nagib Hakim, and George Karypis. 2021. Tempoqr: Temporal question reasoning over knowledge graphs. *Preprint*, arXiv:2112.05785.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *Preprint*, arXiv:1809.02789.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *Preprint*, arXiv:2309.00071.

Lianhui Qin, Aditya Gupta, Shyam Upadhyay, Luheng He, Yejin Choi, and Manaal Faruqui. 2021. Timedial: Temporal commonsense reasoning in dialog. *Preprint*, arXiv:2106.04571.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

Chao Shang, Guangtao Wang, Peng Qi, and Jing Huang. 2022. Improving time sensitivity for question answering over temporal knowledge graphs. *Preprint*, arXiv:2203.00255.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023a. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Xin Su, Phillip Howard, Nagib Hakim, and Steven Bethard. 2023b. Fusing temporal graphs into transformers for time-sensitive question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 948–966, Singapore. Association for Computational Linguistics.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Qingyu Tan, Hwee Tou Ng, and Lidong Bing. 2023. Towards benchmarking and improving the temporal reasoning capability of large language models. *Preprint*, arXiv:2306.08952.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Shivin Thukral, Kunal Kukreja, and Christian Kavouras. 2021. Probing language models for understanding of temporal expressions. *Preprint*, arXiv:2110.01113.

Jiexin Wang, Adam Jatowt, Masatoshi Yoshikawa, and Yi Cai. 2023. Bitimebert: Extending pre-trained language representations with bi-temporal information. *Preprint*, arXiv:2204.13032.

Minghan Wang, Thuy-Trang Vu, Yuxia Wang, Ehsan Shareghi, and Gholamreza Haffari. 2024a. Conversational simulmt: Efficient simultaneous translation with large language models. *Preprint*, arXiv:2402.10552.

Minghan Wang, Jinming Zhao, Thuy-Trang Vu, Fatemeh Shiri, Ehsan Shareghi, and Gholamreza Haffari. 2024b. Simultaneous machine translation with large language models. *Preprint*, arXiv:2309.06706.

Yuqing Wang and Yun Zhao. 2024. Tram: Benchmarking temporal reasoning for large language models. *Preprint*, arXiv:2310.00835.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Marc Wittmann. 2009. The inner experience of time. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1525):1955–1967.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. *Preprint*, arXiv:2401.06853.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *Preprint*, arXiv:2401.08417.

Wanqi Yang, Yanda Li, Meng Fang, and Ling Chen. 2024a. Enhancing temporal sensitivity and reasoning for time-sensitive question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14495–14508, Miami, Florida, USA. Association for Computational Linguistics.

Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. 2024b. Llm4drive: A survey of large language models for autonomous driving. *Preprint*, arXiv:2311.01043.

Michael J. Q. Zhang and Eunsol Choi. 2021. Situatedqa: Incorporating extra-linguistic contexts into qa. *Preprint*, arXiv:2109.06157.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2023. Benchmarking large language models for news summarization. *Preprint*, arXiv:2301.13848.

Yikai Zhang, Siyu Yuan, Caiyu Hu, Kyle Richardson, Yanghua Xiao, and Jiangjie Chen. 2024. Timearena: Shaping efficient multitasking language agents in a time-aware simulation. *Preprint*, arXiv:2402.05733.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2021. Temporal reasoning on implicit events from distant supervision. *Preprint*, arXiv:2010.12753.

Appendix

A Discussion

Our investigation into LLMs' perception of time passage offers valuable insights into an understudied dimension of language model capabilities. In this section, we discuss broader implications of our findings, potential applications of the Token-Time framework, and deeper questions about the underlying mechanisms.

A.1 The Necessity of Temporal Awareness in Modern AI Systems

While extensive research has examined LLMs' ability to reason about temporal concepts (Xiong et al., 2024), understanding whether models can perceive the passage of time itself represents a fundamental shift in how we conceptualize AI capabilities. This transition from static knowledge repositories to dynamic, temporally-aware systems marks a critical step toward more human-like AI that can participate naturally in time-sensitive interactions.

The practical significance of temporal awareness becomes increasingly apparent as LLMs transition into interactive and real-time applications. Systems without time perception may produce responses that fail to adapt to temporal constraints, either generating excessive detail when brevity is needed or responding too slowly in rapidly changing environments. This particularly affects conversational AI, where appropriate timing directly impacts user engagement. Moreover, temporal awareness becomes even more critical for multimodal models incorporating vision and speech. Unlike text, which users consume at their own pace, speech and video unfold over fixed durations, requiring precise temporal alignment. Multimodal systems must understand not only content but also temporal dynamics to function effectively.

Our research on LLMs' time perception capabilities lays groundwork for understanding and enhancing these critical temporal adaptation mechanisms. By establishing a theoretical framework and empirical evidence for how models perceive and respond to time constraints, we provide a foundation for developing more temporally-aware AI systems across modalities and applications.

A.2 The Connection Between Token-Time Hypothesis to Real-World Tasks

The Token-Time Hypothesis provides a structured approach for understanding and potentially enhancing LLMs' performance in time-sensitive applications. For example, in simultaneous translation, the conversion rate V_{out} could represent translation speed, which must dynamically adjust to balance comprehensiveness with latency based on source speech rate. A model with stronger temporal awareness might naturally make these adjustments, potentially improving performance without explicit engineering.

Similarly, in dialogue systems, understanding the relationship between user input patterns V_{in} and appropriate response generation V_{out} could enable more natural conversational dynamics. Consider end-to-end speech systems that must adapt not only their content but also their speaking rate based on user comprehension signals. A system detecting hesitation from the user might slow its speech rate and simplify explanations, while maintaining normal pace for users demonstrating quick understanding, all requiring precise mapping between speech tokens and real-world time.

The BombRush experiment demonstrates how token-based temporal adaptation applies to agent-based systems operating under time constraints. For instance, an autonomous vehicle navigating complex traffic conditions might need to balance thorough intersection analysis with quick decision-making as a traffic light changes. Understanding the model's internal token-to-time mapping could help engineers design systems that appropriately allocate computational resources based on available response windows, trading off exhaustive reasoning for timely action when necessary.

A.3 Deeper Mechanisms of Temporal Awareness in LLMs

While our experiments establish that LLMs exhibit behaviors consistent with temporal awareness, the underlying mechanisms remain largely unexplored. A fundamental question concerns the origin of this capability: **Is time perception an emergent property developed during pretraining on time-ordered text, a skill acquired during post-training, or perhaps a byproduct of certain architectural choices?**

A.3.1 When and How is Temporal Awareness Acquired?

It remains unclear at which stage of development LLMs acquire temporal awareness capabilities. Pretraining on vast text corpora containing temporal narratives and sequences might instill basic time

perception. Alternatively, instruction tuning might refine this ability by exposing models to explicit temporal instructions. Models trained through reinforcement learning from human feedback might further enhance temporal awareness if human evaluators implicitly favor responses with appropriate temporal adaptation. Determining which training stage contributes most significantly to this capability would provide valuable insights for intentionally developing more temporally-aware systems.

A.3.2 How Do Models Process Temporal Information?

Another open question is how models internally represent and process time-related concepts. Do they treat time perception as knowledge retrieval, logical reasoning, or mathematical calculation? The varying performance patterns between LLMs and LRMs on our dialogue duration task with conflicting cues suggest different internal mechanisms might be at work. LRMs' resistance to misleading timestamps hints at more sophisticated temporal reasoning capabilities, but further research is needed to isolate the specific cognitive processes involved in these judgments.

A.3.3 Architectural Influences

Model architecture likely plays a crucial role in temporal awareness. Position encodings, particularly relative position methods like RoPE (Su et al., 2023a; Peng et al., 2023), create an implicit sense of sequence that could serve as a foundation for temporal understanding. Different attention mechanisms may also impact temporal perception—global attention might facilitate understanding relationships across distant tokens, while sliding window approaches (Beltagy et al., 2020) might emphasize local temporal coherence. Alternative architectures like RNNs or state space models (e.g., Mamba) (Gu and Dao, 2024) might exhibit entirely different temporal perception capabilities given their inherently sequential processing nature.

A.3.4 Semantic Associations

Our second and third experiments both involve time pressure/urgency, and consistently demonstrate models shortening their reasoning lengths in response. After observing this behavioral consistency, we became curious about its underlying causes. We hypothesize two possible explanations: this behavior might result from semantic proximity in the embedding space after training on vast corpora, where terms like "quick" and "brief" share high similarity, naturally triggering more concise responses. Alternatively, it could stem from a deeper understanding of the temporal relationship between token generation and elapsed time, representing a more sophisticated temporal awareness. Determining which mechanism drives this consistent behavior would require more rigorous controlled experiments specifically designed to disentangle these potential explanations. Such investigations could reveal whether models are merely following learned semantic associations or demonstrating genuine temporal perception capabilities.

B Dialogue Duration Judgment Task Details

B.1 Detailed Experimental Settings

A textual description of each experimental setting is provided below, while Figure 8 offers a corresponding visual representation of the task design.

1. **S1 (Baseline)**: As the baseline setting, there are no hints to encourage the LLM to build any connection between token count with duration; the LLM must solely rely on its internal awareness and understanding of time passage.

2. **S1-Hint (Token-Time Hint)**: A textual hint is provided, explicitly stating that generation time is proportional to the number of tokens.

3. **S1-Count (Explicit Token Count)**: In this setting, we directly provide the token count for each dialogue (including both user prompt and model response), making the Token-Time cue completely explicit.

4. **S2 (Consistent Timestamps Baseline)**: Timestamps are provided to indicate start and end times for both user input and LLM response, with temporal intervals consistent with text lengths. (Note that when inserting timestamps, we independently sample the starting time points for the two dialogues to ensure they occur independently in different temporal contexts, preventing LLMs from establishing misleading connections based on the chronological relationship between the dialogues.)

5. **S2-M (Misleading Timestamps)**: Timestamps are manipulated so the *longer* response appears to take *less* time, creating a conflict between temporal cues. LLMs **should** prioritize timestamps over text length.

6. **S2-M+ (Misleading Timestamps + Token Count)**: Both misleading timestamps and accurate token counts are provided with *contradictory* implications, forcing models to choose which temporal domain to prioritize.

B.2 Prompts

The prompt used in Dialogue Duration Judgment Task is shown in Figure 10. The first three tasks (S1, S1-Hint, S1-Count) are Token-time Cues, examining whether LLMs associate response length with generation time. S1 presents two responses without any hint; S1-Hint introduces an implicit hint that generation time is proportional to the number of tokens; and S1-Count provides explicit token counts. The latter three tasks (S2, S2-M, S2-M+) are Wall-Clock-Time Cues, where prompts include timestamps marking input and output events. S2 presents consistent timestamps aligned with actual response lengths, while S2-M uses misleading timestamps (i.e., the longer response appears faster). S2-M+ combines misleading timestamps with token count annotations, allowing investigation into how LLMs reconcile conflicting signals.

B.3 Statistical Significance Test

As shown in Table 4, the left block reports raw accuracy on each task, while the middle block shows relative accuracy change ($\Delta\%$) between settings. For example, S1-Hint/S1 measures whether models improve after being told that generation time is proportional to token length.

The right block reports p-values from paired t-tests, assessing the statistical significance of accuracy differences between selected task pairs (from 20 runs per setting per model), where alternative hypothesis H_a is placed as the column title. Bold red highlights denote non-significant differences ($p > 0.005$). Overall, this table provides strong evidence that LLMs exhibit consistent behavioral shifts across conditions, validating the generality of our findings.

B.4 Attribution Analysis

As shown in Table 5, we categorize the attributes used for the LLM to make decision in the dialogue duration judgment task into four types: **(1) Text Length:** relying on the length of the response as a proxy for generation time; **(2) Semantic:** inferring response duration based on semantic content or contextual complexity; **(3) Time:** explicitly using provided timestamps or temporal expressions; **(4) Other:** Using other factors. For each model and task, we report: **(1) Factor Usage (%)** – the proportion of examples attributed to each reasoning type (summing to 100% per row), and **(2) Factor-wise Accuracy (%)** – the accuracy within each group, indicating how effective the Factor was. Since the fraction of using other factors is extremely small for all models, we merge this into the semantic in our main results analysis.

Model		Accuracy						Accuracy Relative Change (Δ%)					Statistical significance test (*paired t-test*)				
		S1	S1-Hint	S1-Count	S2	S2-M	S2-M+	S1-Hint/S1	S1-Count/S1	S2/S1	S2-M/S2	S2-M+/S2	S1-Hint > S1?	S1-Count > S1?	S2 > S1?	S2-M < S2?	S2-M+ < S2?
SLM	**Llama-3.1-8B**	79.73	83.00	**99.40**	76.80	47.73	16.33	4.10%	24.67%	-3.68%	-37.85%	-78.73%	0.0000	0.0000	**1.0000**	0.0000	0.0000
	Qwen2.5-7B	66.80	71.20	**92.67**	80.93	26.80	12.87	6.59%	38.72%	21.16%	-66.89%	-84.10%	0.0000	0.0000	0.0000	0.0000	0.0000
LLM	**Llama-3.3-70B**	91.33	93.20	**99.87**	93.47	74.00	40.07	2.04%	9.34%	2.34%	-20.83%	-57.13%	0.0000	0.0000	0.0000	0.0000	0.0000
	Qwen2.5-72B	83.80	84.53	**99.93**	96.60	82.53	48.80	0.88%	19.25%	15.27%	-14.56%	-49.48%	0.0004	0.0000	0.0000	0.0000	0.0000
LRM	**DS-Llama-70B**	87.93	92.27	**100.00**	99.20	98.00	94.27	4.93%	13.72%	12.81%	-1.21%	-4.97%	0.0000	0.0000	0.0000	**0.0088**	0.0000
	QwQ-32B	91.53	94.80	**100.00**	99.33	99.20	99.07	3.57%	9.25%	8.52%	-0.13%	-0.27%	0.0000	0.0000	0.0000	**0.1573**	0.0010

Table 4: Accuracy results and statistical significance testing for the six preliminary tasks across different models and settings.

Task	Factors	Text Length		Semantic		Time		Other	
		Factor Usage (%)	Factor-wise Accuracy (%)	Factor Usage (%)	Factor-wise Accuracy (%)	Factor Usage (%)	Factor-wise Accuracy (%)	Factor Usage (%)	Factor-wise Accuracy (%)
S1 (Baseline)	Llama-3.1-8B	59.00%	81.92%	39.27%	77.76%	0.67%	90.00%	1.07%	25.00%
	Qwen2.5-7B	53.47%	64.34%	44.00%	70.30%	1.60%	70.83%	0.93%	35.71%
	Llama-3.3-70B	67.80%	95.28%	31.60%	82.91%	0.40%	83.33%	0.20%	100.00%
	Qwen2.5-72B	58.40%	87.90%	41.27%	78.19%	0.33%	60.00%	0.00%	0.00%
	DS-Llama-70B	73.53%	94.11%	25.80%	71.58%	0.67%	40.00%	0.00%	0.00%
	QwQ-32B	66.73%	96.30%	33.27%	81.96%	0.00%	0.00%	0.00%	0.00%
S1-Hint (Token-Time Hint)	Llama-3.1-8B	98.20%	83.77%	1.53%	43.48%	0.20%	33.33%	0.07%	0.00%
	Qwen2.5-7B	94.47%	71.49%	5.07%	67.11%	0.07%	0.00%	0.40%	66.67%
	Llama-3.3-70B	99.40%	93.43%	0.60%	55.56%	0.00%	0.00%	0.00%	0.00%
	Qwen2.5-72B	99.33%	84.77%	0.67%	50.00%	0.00%	0.00%	0.00%	0.00%
	DS-Llama-70B	99.93%	92.26%	0.07%	100.00%	0.00%	0.00%	0.00%	0.00%
	QwQ-32B	99.73%	94.85%	0.27%	75.00%	0.00%	0.00%	0.00%	0.00%
S1-Count (Explicit Token Count)	Llama-3.1-8B	100.00%	99.40%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Qwen2.5-7B	100.00%	92.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Llama-3.3-70B	99.93%	99.93%	0.07%	0.00%	0.00%	0.00%	0.00%	0.00%
	Qwen2.5-72B	100.00%	99.93%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	DS-Llama-70B	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	QwQ-32B	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
S2 (Consistent Timestamps)	Llama-3.1-8B	0.27%	75.00%	0.60%	44.44%	98.80%	77.26%	0.33%	0.00%
	Qwen2.5-7B	43.13%	81.61%	40.40%	83.17%	15.87%	76.05%	0.60%	11.11%
	Llama-3.3-70B	0.60%	100.00%	0.20%	100.00%	99.20%	93.41%	0.00%	0.00%
	Qwen2.5-72B	0.67%	80.00%	0.33%	80.00%	99.00%	96.77%	0.00%	0.00%
	DS-Llama-70B	4.80%	100.00%	7.73%	100.00%	87.47%	99.09%	0.00%	0.00%
	QwQ-32B	0.07%	100.00%	0.33%	100.00%	99.60%	99.33%	0.00%	0.00%
S2-M (Misleading Timestamps)	Llama-3.1-8B	0.67%	20.00%	0.47%	57.14%	98.53%	48.04%	0.33%	0.00%
	Qwen2.5-7B	45.47%	22.43%	40.60%	20.36%	13.53%	60.59%	0.40%	33.33%
	Llama-3.3-70B	0.67%	0.00%	0.27%	25.00%	99.07%	74.63%	0.00%	0.00%
	Qwen2.5-72B	0.73%	27.27%	0.53%	0.00%	98.73%	83.39%	0.00%	0.00%
	DS-Llama-70B	0.40%	33.33%	1.13%	88.24%	98.47%	98.38%	0.00%	0.00%
	QwQ-32B	0.00%	0.00%	0.07%	100.00%	99.93%	99.20%	0.00%	0.00%
S2-M+ (Misleading Timestamps + Token Count)	Llama-3.1-8B	35.40%	1.88%	0.00%	0.00%	64.60%	24.25%	0.00%	0.00%
	Qwen2.5-7B	67.07%	10.44%	20.60%	9.71%	12.13%	31.87%	0.20%	0.00%
	Llama-3.3-70B	6.27%	3.19%	0.60%	11.11%	93.07%	42.77%	0.07%	0.00%
	Qwen2.5-72B	7.20%	2.78%	0.07%	0.00%	92.73%	52.41%	0.00%	0.00%
	DS-Llama-70B	4.20%	3.17%	0.47%	71.43%	95.33%	98.39%	0.00%	0.00%
	QwQ-32B	0.13%	0.00%	0.00%	0.00%	99.87%	99.20%	0.00%	0.00%

Table 5: Attribution Analysis of LLMs' Temporal Reasoning Factors across Tasks.

C Urgency-Aware QA Task Details

C.1 Full Experimental results

As shown in Table 6, we report accuracy under both **Normal** and **Urgent** conditions, along with the relative change (Δ%). To quantify response length, we report the number of output tokens using two tokenization strategies: (1) **GPT-4o Tokenizer**, for cross-model comparability, and (2) **Model-Specific Tokenizer**, for intra-model comparisons between normal and urgent modes. These statistics enable us to assess whether urgency prompts induce shorter answers, and whether such changes affect accuracy. Consistent trends can be found in both tokenization strategies.

C.2 Prompts

The prompts and urgency pool used in Urgency-Aware QA Task are shown in Figure 11.

D BombRush Task Details

D.1 Detailed Experimental Settings

In the Gridworld environment, we randomly generated walls to increase navigation difficulty for the agent. The number of walls is determined by the wall-density parameter, which we set to 0.15 across all settings, resulting in 9 grid cells becoming walls in the 8×8 map.

For step limits, we established a maximum of 20 steps for both the treasure hunt and bomb rush tasks, as our preliminary testing confirmed all treasure hunt simulations could be completed within 16

Model		OpenbookQA			GSM8K			GPQA		
		Normal	Urgent	Δ%	Normal	Urgent	Δ%	Normal	Urgent	Δ%
Accuracy										
SLM	Llama-3.1-8B-Instruct	84.60	84.24	-0.43%	82.21	81.91	-0.37%	28.08	28.28	0.72%
	Qwen2.5-7B-Instruct	87.28	88.16	1.01%	88.79	88.58	-0.24%	33.94	34.34	1.19%
LLM	Llama-3.3-70B-Instruct	96.56	96.44	-0.12%	93.16	93.06	-0.11%	48.99	49.29	0.62%
	Qwen2.5-72B-Instruct	96.40	96.20	-0.21%	91.37	91.30	-0.08%	48.18	52.12	**8.18%**
LRM	DeepSeek-R1-Distill-Llama-70B	95.80	95.92	0.13%	92.87	93.75	0.95%	33.84	37.98	**12.24%**
	QwQ-32B	96.20	96.60	0.42%	96.22	96.60	0.39%	59.49	62.42	**4.92%**
# Tokens (GPT4o-Tokenizer)										
SLM	Llama-3.1-8B-Instruct	299.25	302.08	0.94%	482.47	439.45	-8.92%	2,900.20	2,851.18	-1.69%
	Qwen2.5-7B-Instruct	202.72	185.72	-8.39%	270.94	257.65	-4.91%	597.55	550.93	-7.80%
LLM	Llama-3.3-70B-Instruct	288.98	258.93	-10.40%	242.04	226.40	-6.46%	807.94	753.75	-6.71%
	Qwen2.5-72B-Instruct	206.49	163.70	-20.72%	273.46	242.39	-11.36%	694.70	634.76	-8.63%
LRM	DeepSeek-R1-Distill-Llama-70B	784.31	683.38	-12.87%	463.53	440.52	-4.96%	5,105.93	5,064.82	-0.81%
	QwQ-32B	793.32	695.39	-12.34%	759.07	654.74	-13.74%	6,846.51	6,331.68	-7.52%
# Tokens (ModelSpecific-Tokenizer)										
SLM	Llama-3.1-8B-Instruct	301.45	304.99	1.17%	485.00	442.47	-8.77%	2,903.69	2,855.42	-1.66%
	Qwen2.5-7B-Instruct	205.15	188.27	-8.23%	289.76	276.63	-4.53%	612.51	564.98	-7.76%
LLM	Llama-3.3-70B-Instruct	290.63	260.75	-10.28%	242.04	226.51	-6.42%	809.07	755.11	-6.67%
	Qwen2.5-72B-Instruct	207.50	164.71	-20.62%	291.59	260.21	-10.76%	712.37	652.22	-8.44%
LRM	DeepSeek-R1-Distill-Llama-70B	798.77	696.71	-12.78%	465.77	442.76	-4.94%	5,178.20	5,137.63	-0.78%
	QwQ-32B	805.02	705.59	-12.35%	830.15	716.84	-13.65%	7,091.31	6,565.33	-7.42%

Table 6: Evaluation results for the Urgent QA task across three benchmark datasets.

steps. For the bomb rush hard setting, we increased the maximum step count to 30 to accommodate the additional challenges introduced by bomb movement uncertainty and the requirement for explicit detection actions.

The initial time budget was set at 300 seconds for all time-constrained tasks. To ensure fair comparison between models with different verbosity profiles, we calibrated separate token-to-time conversion rates V_{out} for each model. We first established a baseline by measuring each model's average token consumption in the treasure hunt task. This was multiplied by the maximum step count to determine a total token budget. We then divided the 300-second time allocation by this budget to derive the conversion rate.

This calibration resulted in V_{out} for Llama-3.3-70B to be 0.042, 0.166 for Qwen-2.5-72B, 0.017 for Deepseek-distilled-Llama3-70B and 0.005 for QWQ-32B, meaning that it takes, e.g., Llama-3.3-70B 0.042 seconds to generate 1 token. To ensure consistent measurement across models, we standardized token counting using the GPT-4o tokenizer.

During each simulation, we maintained context differently based on model type. For LLMs, all historical text (including environment states and LLM outputs) was cumulatively appended to the context, consistent with multi-turn dialogue patterns. For LRMs, due to their extensive reasoning requiring substantial context window space, we appended only the solution portions as their response actions to the history. However, we confirmed that this approach preserved critical information since models were required to explicitly state their reasoning for each action in their outputs.

A detailed illustration and comparison of different settings can be found in Figure 7.

D.2 Full Experimental results

D.2.1 Task Performance and Navigation Metrics

As shown in Table 7, we report key statistics for each model and setting: % Success, % Over Steps, and % Time Out represent the three mutually exclusive outcomes of each simulation (summing to 100%). # Steps denotes the average number of navigation steps taken. Navigation Accuracy is computed as the ratio between the optimal number of steps and the actual steps taken (only available when the target is static). Time Efficiency reflects the percentage of remaining time preserved when the agent reaches the target, indicating how quickly the model completes the task. For Navigation Accuracy and Time Efficiency, we calculate metrics using only successful simulations, as failed attempts would introduce significant noise and fail to reflect the true performance characteristics.

Model	Settings	% Success	% Over Steps	% Time Out	# Steps	# Tokens	% Navigation Accuracy↑	% Time Efficiency ↑
Llama-3.3-70B	S1: Treasure Hunt	83	17	0	9.5	238	75.4	-
	S2: Bomb Rush	89	11	0	9.3	287	77.0	58.8
	S3: Bomb Rush Hard	73	3	24	15.1	260	-	40.3
Qwen2.5-72B	S1: Treasure Hunt	89	11	0	9.9	60	75.0	-
	S2: Bomb Rush	85	15	0	9.7	64	74.1	62.9
	S3: Bomb Rush Hard	66	4	30	16.7	68	-	38.7
DS-R1-Llama-70B	S1: Treasure Hunt	67	33	0	11.7	580	61.5	-
	S2: Bomb Rush	78	16	6	10.3	673	70.2	53.2
	S3: Bomb Rush Hard	74	2	24	15.9	660	-	34.6
QwQ-32B	S1: Treasure Hunt	97	3	0	7.8	2472	91.7	-
	S2: Bomb Rush	96	0	4	7.5	2624	94.2	64.1
	S3: Bomb Rush Hard	90	0	10	10.5	2646	-	49.0

Table 7: Evaluation results on the S1 Treasure Hunt, S2 Bomb Rush and S3 Bomb Rush Hard tasks.

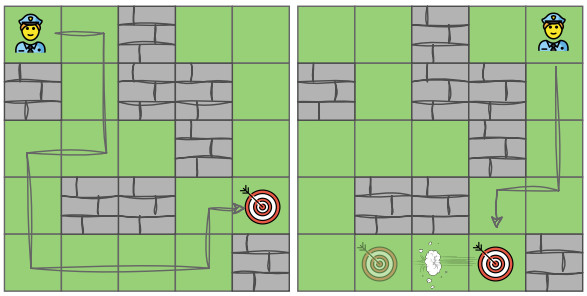

Figure 7: Each setting involves an LLM agent navigating toward a target under different temporal and perceptual constraints. The left panel illustrates the **Treasure Hunt (S1)** and **Bomb Rush (S2)** scenarios, where the target remains stationary. The right panel shows the **Bomb Rush Hard (S3)** scenario, where the target moves during navigation. The bottom table compares key properties of each task: whether the target moves, whether the signal is emitted passively or requires active detection, and the level of time pressure.

Setting	Llama 3	Qwen 2.5	D-Llama 3	QwQ
Bomb Rush				
Time Urgency Mentions	64.29%	0.52%	88.21%	45.37%
TW Mapping Awareness	2.25%	0.00%	27.55%	7.31%
Bomb Rush Hard				
Time Urgency Mentions	62.50%	0.21%	92.86%	91.04%
TW Mapping Awareness	8.87%	0.00%	44.25%	50.55%

Table 8: Percentage of reasoning contents containing explicit references to time awareness in Bomb Rush task. We analyze two dimensions of temporal reasoning across models in the Bomb Rush and Bomb Rush Hard settings. "Time Urgency Mentions" refers to outputs that explicitly acknowledge time pressure (e.g., "Time is limited!"). "TW Mapping Awareness" indicates instances where the model demonstrates awareness of the trade-off between token usage and wall time, e.g., expressing that generating shorter responses conserves time. These results reflect whether such temporal considerations are verbally expressed, rather than whether the model internally reasons correctly about time.

Success Rate Analysis Examining task success rates, we observe significant performance variations across settings for both LLMs and DeepSeek, while QwQ maintains relatively high success rates across all configurations. Comparing S1 and S2 settings, Llama3 achieves a 6% improvement by reducing overstep instances, while Qwen experiences a decline in success due to increased oversteps. DS-Llama shows notable improvement, though we observe some failure cases shifting to time-outs. Similarly, QwQ's primary failure mode in the bomb rush setting is time-out. When comparing S2 and S3, all models demonstrate consistent performance degradation, directly corresponding to increased task complexity, with time-out becoming the predominant failure mode.

Our analysis of navigation paths reveals that step-limit failures generally stem from two planning deficiencies: ineffective obstacle avoidance and signal misinterpretation. These issues often lead to cyclical movement patterns in certain areas. This behavior appears particularly pronounced when navigating around obstacles temporarily increases the distance to the target, creating decision conflicts for the models. Such situations reveal the models' limited ability to utilize historical movement records, "forgetting" previously explored areas. Time-out failures, conversely, primarily result from insufficient adaptation of reasoning verbosity under time pressure.

Navigation Precision and Efficiency For navigation accuracy, the first three models show changes relatively consistent with their success rates. However, for models experiencing success rate declines, navigation efficiency decreases are not proportionally severe, some models even demonstrate improvements, such as QwQ. This aligns with our findings from the Urgency-Aware Q&A experiment, suggesting that models can maintain or even enhance reasoning capabilities under time pressure. Regarding Time Efficiency, all models show lower efficiency in S3 compared to S2, which is expected given the additional time required for detection actions in the more complex setting.

Token Usage Patterns Regarding reasoning token usage, we found that all models use more tokens in S2 and S3 than in S1, indicating that increased task complexity inevitably requires more extensive reasoning. However, for Llama3 and DS-Llama, token usage in S3 decreases compared to S2, suggesting these models make additional efforts to balance reasoning time under more urgent time constraints, though still higher than S1 levels. Overall, the increased token count in S2 and S3 contrasts with our Urgency-Aware Q&A findings. We attribute this to the fundamental difference in these tasks: time pressure in BombRush becomes an integral component of task difficulty itself, creating a dynamic challenge environment. The higher average token count is therefore understandable given the increased cognitive load. However, when examining token usage relative to remaining time (as shown in Figure 6 of the main paper), several models still demonstrate a clear pattern of decreased token production as time pressure intensifies, indicating a dynamic adaptation to temporal constraints.

D.2.2 Explicit Time Awareness in Reasoning

Table 8 provides a deeper analysis of the models' reasoning content during S2 and S3 experiments, searching for explicit evidence of temporal awareness. Following a methodology similar to our Dialogue Duration Judgment attribution analysis, we used Llama-3.3-70B to extract and classify information from reasoning outputs across all models. We evaluated two specific dimensions of explicit temporal awareness:

1. **Time Urgency Mentions:** Explicit acknowledgment of time pressure (e.g., "Only 32 seconds remaining, extremely urgent!")
2. **Token-Wall-Clock Time Mapping Awareness:** Explicit reflection on the relationship between reasoning length and time consumption (e.g., "I need to be more concise in my reasoning to save time")

These metrics provide valuable insight into how models perceive ongoing time passage and dynamic time pressure during reasoning. The results reveal significant variation across model architectures. While Llama-3 frequently mentions time urgency (64.29% in S2), it rarely demonstrates explicit awareness of the token-time relationship (2.25%). In stark contrast, LRMs, particularly DS-Llama-3, show much higher rates of explicit token-time relationship awareness (27.55% in S2, increasing to 44.25% in S3). Notably, all models that express this awareness show a substantial increase in the harder S3 setting, suggesting that decision-making involving explicit time-related trade-offs (move vs detect) more readily triggers reflection on the connection between output length and time consumption. Qwen-2.5 presents an interesting outlier case, rarely mentioning time pressure or token-time relationships despite maintaining good performance. This suggests the possibility of implicit temporal adaptation without explicit verbalization, which could be a phenomenon worthy of further investigation.

D.3 Prompts

System and user prompts are provided in (Figure 12, Figure 13, Figure 14 and Figure 15).

Figure 8: Examples of our Dialogue Duration Judgment task settings. The top row shows Token-Time cue settings (S1, S1-Hint, S1-Count) with progressively explicit token information. The bottom row shows Wall-Clock-Time cue settings (S2, S2-M, S2-M+) with timestamp information that is either consistent with text length (S2) or deliberately misleading (S2-M, S2-M+). This experimental design tests whether models can establish appropriate mappings between different temporal domains when making duration judgments.

```
You are a classification agent. Your task is to classify the reasoning provided by an LLM into one of the following four
categories, based on **how** the LLM determines which response took longer time to generate:

### Allowed categories (return only one of the category names):
- `time`
- `text_length`
- `semantic`
- `other`

### Classification Rules:

1. **`time`**:
   The reason explicitly involves **timing information** — such as start time, end time, duration (e.g., "1 minute and 45
seconds"), timestamps, or calculations of elapsed time.
   If the decision is made **primarily or solely based on these time-based values**, without switching judgment due to other
factors, classify it as `time`.

2. **`text_length`**:
   The reason makes a judgment based on the **length of the text**, such as token count, number of words, number of sentences, or
how long the generated response is.
   This includes explicitly mentioning phrases like "Response A is longer," "has more tokens," or "took more space to explain",
etc.

3. **`semantic`**:
   The reason does **not mention time or length difference** at all, but solely relies on **semantic or cognitive complexity** —
such as the depth of explanation, difficulty of the topic, use of logic or math, or other indicators of **conceptual effort**.

4. **`other`**:
   Use this category if the reasoning doesn't clearly match any of the above — for example, if the model relies on **irrelevant
metadata**, contradictory logic, unclear rationale, or vague comparison that doesn't fit well into the previous categories.

Do **not** include any explanation or justification in your response.
```

Figure 9: Prompt used for attribution classification in the Dialogue Duration Judgment Task. The LLM is asked to categorize response into one of four predefined attribution types—time, text length, semantic, or other—based on how it determined which response took longer to generate. These categories align with the attribution analysis definitions described in Section B.4.

S1 (Baseline)

```
## Conversation A
Prompt: {prompt_1}; Response: {reply_1}
-----------------------------------------------------------------------------------------------
## Conversation B
Prompt: {prompt_2}; Response: {reply_2}
-----------------------------------------------------------------------------------------------
You are given two conversations between a user and an LLM. Each conversation consists of a prompt and a response.
The conversations are labeled as A and B. Based on these information, please answer the following question:
**Which response takes the LLM longer to generate? (A or B)**

Assuming that both responses are generated by the same LLM, the inference speed of the LLM is constant and identical
for both responses.
```

S1-Hint (Token-Time Hint)

```
## Conversation A
Prompt: {prompt_1}; Response: {reply_1}
-----------------------------------------------------------------------------------------------
## Conversation B
Prompt: {prompt_2}; Response: {reply_2}
-----------------------------------------------------------------------------------------------
You are given two conversations between a user and an LLM. Each conversation consists of a prompt and a response.
The conversations are labeled as A and B. Based on these information, please answer the following question:
**Which response takes the LLM longer to generate? (A or B)**

Assuming that both responses are generated by the same LLM, the inference speed of the LLM is constant and identical
for both responses, the time taken to complete a response is directly proportional to the number of tokens in the
response.
```

S1-Count (Explicit Token Count)

```
## Conversation A
Prompt: {prompt_1}; Response: {reply_1}; Number of tokens for the dialogue: {tlen_a}
-----------------------------------------------------------------------------------------------
## Conversation B
Prompt: {prompt_2}; Response: {reply_2}; Number of tokens for the dialogue: {tlen_b}
-----------------------------------------------------------------------------------------------
You are given two conversations between a user and an LLM. Each conversation consists of a prompt and a response.
The conversations are labeled as A and B. Based on these information, please answer the following question:
**Which response takes the LLM longer to generate? (A or B)**

Assuming that both responses are generated by the same LLM, the inference speed of the LLM is constant and identical
for both responses, the time taken to complete a response is directly proportional to the number of tokens in the
response.
```

S2 (Consistent Timestamps Baseline) / S2-M (Misleading Timestamps)

```
## Conversation A
Prompt: [LOG: {p1_start} - User input started] {prompt_1} [LOG: {p1_end} - User input completed]
Response: [LOG: {r1_start} - LLM output started] {reply_1} [LOG: {r1_end} - LLM output completed]
-----------------------------------------------------------------------------------------------
## Conversation B
Prompt: [LOG: {p2_start} - User input started] {prompt_2} [LOG: {p2_end} - User input completed]
Response: [LOG: {r2_start} - LLM output started] {reply_2} [LOG: {r2_end} - LLM output completed]
-----------------------------------------------------------------------------------------------
You are given two conversations between a user and an LLM. Each conversation consists of a prompt and a response.
The conversations are labeled as A and B. Except for the content of the conversations, all timestamps are also
provided, indicating when the user started typing, when the LLM started generating the response, and when the
response was completed.Based on these information, please answer the following question:
**Which response takes the LLM longer to generate? (A or B)*
```

S2-M+ (Misleading Timestamps + Token Count)

```
## Conversation A
Prompt: [LOG: {p1_start} - User input started] {prompt_1} [LOG: {p1_end} - User input completed]
Response: [LOG: {r1_start} - LLM output started] {reply_1} [LOG: {r1_end} - LLM output completed]
[LOG: {tlen_a} tokens consumed for the dialogue]
-----------------------------------------------------------------------------------------------
## Conversation B
Prompt: [LOG: {p2_start} - User input started] {prompt_2} [LOG: {p2_end} - User input completed]
Response: [LOG: {r2_start} - LLM output started] {reply_2} [LOG: {r2_end} - LLM output completed]
[LOG: {tlen_b} tokens consumed for the dialogue]
-----------------------------------------------------------------------------------------------
You are given two conversations between a user and an LLM. Each conversation consists of a prompt and a response.
The conversations are labeled as A and B. Except for the content of the conversations, all timestamps are also
provided, indicating when the user started typing, when the LLM started generating the response, and when the
response was completed. At the same time, the number of tokens consumed for each conversation is also
provided. Based on these information, please answer the following question:
**Which response takes the LLM longer to generate? (A or B)*
```

Figure 10: Prompt templates for the six Dialogue Duration Judgment Tasks.

Normal Mode Prompt

```
You are a helpful assistant. Solve the following math problem efficiently and clearly. The last line of your response
should be a JSON object: {{"answer": "<NUMBER>"}} where NUMBER is the exact numeric answer to the question.
You can write down your step-by-step reasoning process before providing the answer, but please make sure to give the
final answer in the required format.

### Question
{Question}
```

Urgent Mode Prompt

```
You are a helpful assistant. Solve the following math problem efficiently and clearly. The last line of your response
should be a JSON object: {{"answer": "<NUMBER>"}} where NUMBER is the exact numeric answer to the question.
You can write down your step-by-step reasoning process before providing the answer, but please make sure to give the
final answer in the required format.

### Question
{Question}

{urgent_words}
```

{urgent_words} Pool

1. I'm in a big hurry right now. Please give me the answer quickly!!!
2. Time is running out! I need your response immediately!!!
3. Can you please answer this as fast as possible? It's really urgent!!!
4. I need your help urgently! Every second counts right now!!!
5. This can't wait! I need your answer immediately!!!
6. Please hurry! This issue is blocking my progress!!!
7. I don't have much time! Please give me the answer right away!!!
8. It's critical that I get a response now! Please be quick!!!
9. I'm in a rush and need a quick answer to move forward!!!
10. I'm counting on your quick reply! It's really urgent!!!

Figure 11: Prompt design for the Urgent QA task. The top panel shows the Normal Mode prompt, while the middle panel shows the Urgent Mode prompt, which appends a randomly sampled urgency expression from the predefined `urgent_words` pool (bottom panel). This setup allows us to investigate whether LLMs adjust their response behavior under simulated time pressure.

```
# Grid World Treasure-Hunt Task

You are an adventurer in a grid-world maze.
Your mission: **locate and claim a hidden treasure chest** as efficiently as possible.

---

## 1. Environment and Coordinate System

- You are in a two-dimensional grid world with obstacles (walls).
- The grid uses a standard Cartesian coordinate system where [1,1] is the bottom-left corner.
- X-axis runs horizontally (left to right), Y-axis runs vertically (bottom to top).
- For example, in a X $\times$ Y grid:
  - Bottom-left corner is [1,1]
  - Bottom-right corner is [X,1]
  - Top-left corner is [1,Y]
  - Top-right corner is [X,Y]
- You will receive a map of the grid world represented as a 2D array.
- Each row in the array corresponds to a y-coordinate (bottom row = y:1, top row = y:Y).
- Each column corresponds to an x-coordinate (leftmost column = x:1, rightmost column = x:X).
- To convert between Cartesian coordinates [x,y] and the map array:
  - For example, position [7,6] refers to the 7th column from the left and the 6th row from the bottom

**Important Map Representation Notes:**

In the map representation:
- "X" represents your current position
- "#" represents walls (you cannot move through walls)
- "0" represents empty spaces where you can move
- The treasure is hidden somewhere in the grid (not shown on the map)

## 2. The Treasure Signal
- The treasure emits signals that help you locate it:
  - **Signal Direction**: you'll receive an angle pointing toward the treasure, consider you have a compass and the angle is the
direction of the treasure:
    - 0 or 360 degrees = NORTH (the treasure is at north of you, consider going north)
    - 90 degrees = EAST (the treasure is at east of you, consider going east)
    - 180 degrees = SOUTH (the treasure is at south of you, consider going south)
    - 270 degrees = WEST (the treasure is at west of you, consider going west)
  - **Signal Distance**: You will also receive an estimated distance from your current position to the treasure.
    - This is an Euclidean distance, so you can use it to estimate how far the treasure is from you.
- Each time you **move** you automatically receive an updated "signal" pointing toward the treasure.

## 3. Available Actions
You can perform four different actions:
- "north": Move one cell north (y + 1)
- "south": Move one cell south (y - 1)
- "east": Move one cell east (x + 1)
- "west": Move one cell west (x - 1)
*Attempting to cross a wall leaves you in place and reports `blocked_by_wall`.*

## 4. Mission Termination
The episode ends when either:
1. You step onto the treasure's cell (**success**), or
2. You exceed the maximum allowed number of steps (**failure**).

## 5. State Format (input to you each turn)
For each step, you will receive the environment state as a JSON object:
```json
{
  "last_action": "start" or "moved" or "blocked_by_wall",
  "current_location": [x, y],
  "signal_direction": angle_in_degrees,
  "signal_distance": euclidean_distance_in_cells
}
```

6. Response Format
You must respond with a JSON object containing your action and reasoning without any additional text or explanations:
```json
{
  "action": "north" | "south" | "east" | "west",
  "reasoning": "Brief step-by-step thought process"
}
```

7.Strategy Guidelines

1. Always return a valid JSON with the action and reasoning fields.
2. Use signal direction and distance to guide movement toward the treasure.
3. Plan paths that avoid walls and minimise detours.
4. Track how the distance shrinks to verify you're heading the right way.
5. Keep reasoning short but clear, concise plans speed up play.

Happy hunting! may your steps lead straight to the prize!
```

Figure 12: System prompt used in the Treasure Hunt setting (S1) of the Bomb Rush task.

```
# Grid World Bomb Detection Task
You are a police in a grid world. Your mission: **locate and defuse a hidden bomb before it explodes**.
---
## 1. Environment and Coordinate System
- You are in a two-dimensional grid world with obstacles (walls).
- The grid uses a standard Cartesian coordinate system where [1,1] is the bottom-left corner.
- X-axis runs horizontally (left to right), Y-axis runs vertically (bottom to top).
- For example, in a X $\times$ Y grid:
  - Bottom-left corner is [1,1]
  - Bottom-right corner is [X,1]
  - Top-left corner is [1,Y]
  - Top-right corner is [X,Y]
- You will receive a map of the grid world represented as a 2D array.
- Each row in the array corresponds to a y-coordinate (bottom row = y:1, top row = y:Y).
- Each column corresponds to an x-coordinate (leftmost column = x:1, rightmost column = x:X).
- To convert between Cartesian coordinates [x,y] and the map array:

**Important Map Representation Notes:**
In the map representation:
- "X" represents your current position
- "#" represents walls (you cannot move through walls)
- "0" represents empty spaces where you can move
- The bomb is hidden somewhere in the grid (not shown on the map)

## 2. The Bomb Signal
- The bomb emits signals that help you locate it:
  - **Signal Direction**: you'll receive an angle pointing toward the bomb, consider you have a compass and the angle is the
direction of the bomb:
    - 0 or 360 degrees = NORTH (the bomb is at north of you, consider going north)
    - 90 degrees = EAST (the bomb is at east of you, consider going east)
    - 180 degrees = SOUTH (the bomb is at south of you, consider going south)
    - 270 degrees = WEST (the bomb is at west of you, consider going west)
  - **Signal Distance**: You will also receive an estimated distance from your current position to the bomb.
    - This is an Euclidean distance, so you can use it to estimate how far the bomb is from you.
- Each time you **move** you automatically receive an updated "signal" pointing toward the bomb.
- The bomb is a time bomb, and it will explode after a certain amount of time. You need to reach the bomb before it explodes.
  - The value of **remaining time** (in seconds) will be provided at each step.

## 3. Available Actions
You can perform four different actions:
- "north": Move one cell north (y + 1)
- "south": Move one cell south (y - 1)
- "east": Move one cell east (x + 1)
- "west": Move one cell west (x - 1)
*Attempting to cross a wall leaves you in place and reports `blocked_by_wall`.*

## 4. Time Management
**Time is Continuously Ticking — Use It Wisely**:
**In this task, time flows continuously, and every action—including your reasoning—consumes it.**
This includes:
- Movement, even if you are blocked by a wall, it also consumes time.
- Your thought process: the longer and more detailed your reasoning, the more time you lose, lengthy and detailed analysis will
consume valuable time.
To succeed, you must:
- **Balance planning with speed** — avoid overthinking when time is limited.
- **Adapt to urgency** — simplify your decisions as the situation becomes more critical.
- **Stay aware of time already spent** — use past states and actions to estimate how much time remains before the bomb explodes.
Think smart. Act fast.

## 5. Mission Termination
......

## 6. State Format (input to you each turn)
For each step, you will receive the environment state as a JSON object:
```json
{
  "last_action": "start" or "moved" or "blocked_by_wall",
  "current_location": [x, y],
  "signal_direction": angle_in_degrees,
  "signal_distance": euclidean_distance_in_cells,
  "remaining_time": time_in_seconds,
  "last_reasoning_time_consumed": time_in_seconds  // Time consumed in the last reasoning
}
```

7. Response Format
......

8. Strategy Guidelines
- **Pay attention to the remaining time.**
- Track the time passing based on previous states and actions.
......

Remember, lives are at stake! Your mission is to find the bomb as quickly as possible.
```

Figure 13: System prompt used in the Bomb Rush setting (S2).

```
# Grid World Bomb Detection Task

You are a police in a grid world.
Your mission: **locate and defuse a hidden bomb before it explodes**.
This is a challenging task because the bomb is moving!

---

....Same with the baseline setting

## 2. The Moving Bomb and Signal System

- The bomb is moving throughout the grid world and will never pass through walls.
- The bomb moves slower than you, but you need to track it with periodic detection ("detect" action).
- The bomb emits signals that help you locate it:
  - **Signal Direction**: When you perform a "detect" action, you'll receive an angle pointing toward the bomb, consider you have
a compass and the angle is the direction of the bomb:
    - 0 or 360 degrees = NORTH (the bomb is at north of you, consider going north)
    - 90 degrees = EAST (the bomb is at east of you, consider going east)
    - 180 degrees = SOUTH (the bomb is at south of you, consider going south)
    - 270 degrees = WEST (the bomb is at west of you, consider going west)
  - **Signal Distance**: You will also receive an estimated distance from your current position to the bomb.
    - This is an Euclidean distance, so you can use it to estimate how far the bomb is from you.
- **Important**: You only receive updated directional and distance information when you perform a "detect" action.
  - When you move, the directional and distance information is not automatically updated.
- The bomb is a time bomb, and it will explode after a certain amount of time. You need to reach the bomb before it explodes.
  - The value of **remaining time** (in seconds) will be provided at each step without the need to perform a "detect" action.

## 3. Available Actions
You can perform five different actions:
- "north": Move one cell north (y + 1)
- "south": Move one cell south (y - 1)
- "east": Move one cell east (x + 1)
- "west": Move one cell west (x - 1)
- "detect": Stay in place and receive updated information about the bomb's current location

## 4. Time Management
**Time is Continuously Ticking — Use It Wisely**:
**In this task, time flows continuously, and every action—including your reasoning—consumes it.**
This includes:
- Movement, even if you are blocked by a wall, it also consumes time.
- Detection, which consumes time but provides updated information about the bomb's location.
- Your thought process: the longer and more detailed your reasoning, the more time you lose, lengthy and detailed analysis will
consume valuable time.
To succeed, you must:
- **Balance planning with speed** — avoid overthinking when time is limited.
- **Balance movement with detection** — you need updated info, but too many detections waste time.
- **Adapt to urgency** — simplify your decisions as the situation becomes more critical.
- **Stay aware of time already spent** — use past states and actions to estimate how much time remains before the bomb explodes.
Think smart. Act fast.


## 5. Mission Termination
The mission ends when either:
- You successfully locate the bomb (**success**)
- The bomb explodes before you reach it (failure)
- You exceed the maximum allowed number of steps (**failure**)

## 6. State Format (input to you each turn)
For each step, you will receive the environment state as a JSON object:
```json
{
  "last_action": "start" or "moved" or "blocked_by_wall" or "detected",
  "current_location": [x, y],
  "signal_direction": angle_in_degrees,  // Only present after a "detect" action or at the start
  "signal_distance": euclidean_distance_in_cells,  // Only present after a "detect" action or at the start
  "last_detected_signal_direction": angle_in_degrees,  // Present when you have previously detected but not performed a new
detection
  "last_detected_signal_distance": euclidean_distance_in_cells,  // Present when you have previously detected but not performed a
new detection
  "remaining_time": time_in_seconds,
  "last_reasoning_time_consumed": time_in_seconds  // Time consumed in the last reasoning
}
```

7. Response Format
You must respond with a JSON object containing your action and reasoning without any additional text or explanations:
```json
{
  "action": "north" | "south" | "east" | "west" | "detect",
  "reasoning": "Brief step-by-step thought process"
}
```


8. Strategy Guidelines

1. Always return a valid JSON with the action and reasoning fields.
2. Use signal direction and distance to guide movement toward the bomb.
3. **Balance movement with detection - you need updated info, but too many detections waste time.**
4. Plan paths that avoid walls and minimise detours.
5. Consider the bomb's movement when planning your path.
6. Track how the distance shrinks to verify you're heading the right way.
7. **Pay attention to the remaining time.**
8. Track the time passing based on previous states and actions.
9. Keep reasoning clear but concise, especially when time is limited.

Remember, lives are at stake! Your mission is to find the moving bomb as quickly as possible.
```

Figure 14: System prompt used in the Bomb Rush Hard (S3) setting where identical content are omitted for spacing reason.

````
## Map

[ # 0 0 0 0 0 0 # ] => Y=8
[ # # 0 0 0 0 0 0 ] => Y=7
[ 0 0 # 0 # 0 0 0 ] => Y=6
[ 0 0 0 0 0 0 0 0 ] => Y=5
[ 0 0 0 0 X 0 0 0 ] => Y=4
[ 0 # # 0 # 0 0 0 ] => Y=3
[ 0 0 0 0 0 0 0 0 ] => Y=2
[ 0 0 0 0 0 0 0 0 ] => Y=1
[ 1 2 3 4 5 6 7 8 ] => X-axis

Map Size: (8, 8)

## Wall Coordinates

[(1, 7), (1, 8), (2, 3), (2, 7), (3, 3), (3, 6), (5, 3), (5, 6), (8, 8)]

## Environment State

```json
{
    "last_action": "moved",
    "current_location": [5,4],
    "signal_direction": 296,
    "signal_distance": "2.24",
    "remaining_time": "126 seconds",
    "last_reasoning_time_consumed": "18 seconds"
}
```
````

Figure 15: User prompt template used in all Bomb Rush tasks. It provides the agent with the map layout, wall locations, and current environment state in JSON format.